Конспект лекций по ПОЦОД

2 лекция. Методология разработки ПО

Жизненный цикл ПО – период времени от возникновения идеи создания ПО до вывода его из эксплуатации

Был ГОСТ 34.601-90:
Формирование требований к АС → Разработка концепции АС → Техническое задание → Эскизный проект → Технический проект → Рабочая документация → Ввод в действие → Сопровождение АС

ISO/IEC 12207:2008 – Специальные процессы программных средств
Процесс анализа требований ПС → Процесс проектирования архитектуры ПС → Процесс детального проектирования ПС → Процесс конструирования ПС → Процесс квалификационного тестирования ПС → Процесс комплексирования ПС

Процессы поддержки ПС:
- Процесс менеджмента документации
- Процесс менеджмента конфигурации
- Процесс обеспечения характеристик качества
- Процесс решения проблем в ПС
- Процесс верификации ПС
- Процесс аудита ПС
- Процесс ревизии ПС
- Процесс валидации ПС

Валидация – подтверждение того, что требования, предназначенные для конкретного использования или применения, выполнены

Верификация – подтверждение того, что заданные требования полностью выполнен

Аудит – независимая оценка программных продуктов и процессов, проводимая уполномоченным лицом с целью оценить их соответствие требованиям

Ревизия – процесс, проводимый с целью составления и поддержки общего с правообладателями понимания относительно целей ~~проекта~~ соглашения и того, что именно необходимо сделать для помощи в обеспечении разработки продукта, удовлетворяющего правообладателя

Правообладатель – сторона, причастная к программному продукту или заинтересованная в нём, имеющая права, долю, ~~требования~~ или интересы относительно продукта или его свойств

Пример жизненного цикла программного продукта:
Определение требований (стейкхолдеры, аналитики) → Проектирование (аналитики, архитекторы, UI-дизайнеры) → Разработка (разработчики) → Тестирование (тестировщики) → Документация (технические писатели) → Поставка & Сопровождение (Администраторы, техническая поддержка)

Методология разработки ПО – структура, согласно которой построена разработка ПО
Основные группы методологий разработки:
- Каскадная модель
- Итеративная модель
- Спиральная модель

Водопадная модель ЖЦ предусматривает последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе
Этапы проекта в соответствии с каскадной моделью:
- Формирование требований
- Проектирование
- Реализация
- Тестирование
- Внедрение
- Эксплуатация и сопровождение

Выработка системных требований → Выработка требования к ПО → Анализ → → Проектирование → Кодирование → Тестирование → Эксплуатация

Преимущества:
- Полная документация на каждом этапе
- Строгость требований и их неизменность
- Жёсткие сроки и затраты на проект

↓

Недостатки:

- В классическом виде не работает в реальной жизни
- Сложность исправления ошибок предыдущих этапов
- Сложность внесения изменений в требования после старта проекта — негибкость

Спиральная методология применяется в том случае, если цена ошибки на ранних этапах очень высока или есть высокий риск изменения требований после старта проекта

4 этапа для каждого витка:

- Планирование
- Анализ рисков
- Конструирование
- Оценка результата и при удовлетворительном качестве переход к новому витку

В целом более или менее рабочая версия продукта получается только на финальном витке, но при этом анализ рисков на ранних витках позволяет избежать обнаружения проблем только на последнем этапе

Итеративная модель разработки ПО — выполнение работ происходит параллельно с непрерывным анализом полученных ранее результатов и корректировкой предыдущих этапов работы

Планирование → Реализация → Проверка → Оценка

Преимущества:

- Возможность внесения изменений в требования к продукту после старта проекта, часто — эффективная обратная связь от стейкхолдеров
- Снижение влияния ошибок на ранних этапах на сроки и стоимость проекта
- Более равномерная загрузка участников проекта
- Эффективное использование накопленного опыта

Недостатки:

- Отсутствие жёстких требований, их рассогласованность

Инкрементная модель — с каждой итерацией появляется подсистема общей системы в рабочем состоянии.

Rational Unified Process — итеративная методология разработки ПО, созданная компанией Rational Software

Принципы:

- Ожидание изменений в требованиях, проектных решениях и реализации в процессе разработки
- Прототипы, реализуемые и тестируемые на ранних этапах проекта
- Постоянное тестирование на этапах разработки проекта
- Ранняя идентификация и непрерывное устранение основных рисков
- Концентрация на выполнении требований заказчиков к исполняемой программе

Преимущества:

- Возможность учитывать изменения требований
- Масштабируемость

Гибкие методологии разработки ПО — серия подходов к разработке ПО, основанных на:

- Итеративной разработке
- Динамическом формировании и постоянном изменении требований
- Взаимодействии внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля

Agile Manifesto:

- Люди и взаимодействие важнее процессов и инструментов
- Работающий продукт важнее исчерпывающей документации
- Сотрудничество с заказчиком важнее согласования условий контракта
- Готовность к изменениям важнее следования первоначальному плану

Agile методологии:

- Scrum
- XP - экстремальное программирование
- Crystal
- Kanban
- Lean

Scrum — набор принципов, на которых строится процесс разработки:

- Жёстко фиксированные и небольшие по времени итерации, называемые спринтами
- Регулярное предоставление конечному пользователю работающего ПО с новыми возможностями
- Самоорганизация команды, состоящей из небольшого количества специалистов

Спринт — итерация разработки, в итоге которой реализуется значимое изменение функциональности ПО. Длительность - 1-4 недели

Задачи в проекте:

- Бэклог — список всех задач, которые нужно реализовать
- Задачи спринта — задачи, решаемые только в данном спринте. Список фиксируется в начале спринта. Если что-то не сделано к концу спринта — переходит в следующий спринт

Основные роли:

- Scrum-master — проводит встречи, следит за соблюдением принципов Scrum. Чаще всего совмещена с другой ролью
- Product owner — представляет интересы стейкхолдеров
- Team — небольшая (до 10 чел.) команда специалистов различных профилей (аналитики, дизайнеры, разработчики, тестировщики, администраторы и сопровождение)

Дополнительные:
- Клиенты
- Менеджеры проекта
- Sales-менеджеры
- Консультанты

Встречи в рамках спринта:
- Sprint planning – проводится перед началом очередного спринта, команда обсуждает и выбирает, какие задачи будут реализованы в спринте
- Daily meeting – ежедневная короткая встреча, на которой каждый из членов команды рассказывает о своей работе
- Sprint review – проводится по окончании спринта и посвящена тому, что было сделано командой
- Retrospective – встреча, на которой присутствуют только члены команды, посвящена проблемам, возникшим в ходе спринта, усовершенствованию процесса взаимодействия внутри и вне команды

Экстремальное программирование – заимствование традиционных техник программирования, но приведение их к некоторому экстремальному варианту:
- Очень короткие циклы
- Коллективное владение кодом → парное программирование
- "Заказчик всегда рядом"
- Непрерывная интеграция – рабочие ветки сливаются в общую несколько раз в день, постоянные сборки версий
- Разработка через тестирование

Принципы Crystal:
- Человеко-ориентированные, а не процессо-ориентированные методологии
- Индивидуальный подбор методологии под проект и команду
- Максимально простые методологии

Lean Development – принципы разработки ПО, возникшие из принципов бережливого производства:
- Уменьшение потерь (потери – то, что не добавляет ценности продукта для пользователя)
- Непрерывное обучение и совершенствование
- Поздние принятие решений
- Максимально быстрая доставка пользователю
- Определение потока ценностей
- Мотивация

Канбан – метод управления разработкой, реализующий принцип «точно в срок» и способствующий равномерному распределению нагрузки между работниками

↓

Принципы:
- Визуализация разработки
- Разделение работы на задачи
- Использование отметок отложенных задач в разработке
- Ограничение работ, выполняющихся одновременно, на каждом этапе разработки

Менеджмент проекта – деятельность по достижению поставленной цели, осуществляемая методом балансирования:
- Объёма работ
- Ресурсов (деньги, люди)
- Времени выполнения проекта
- Качества продукта

Риск-менеджмент – процесс выявления, оценки и снижения вероятности событий, которые могут негативно влиять на результат проекта

---

4 лекция. Тестирование ПО

Тестирование ПО – проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранных определённым образом

Тестирование ПО – один из этапов его разработки

Определение требований → Проектирование → Разработка → Тестирование → Документация → Поставка → Сопровождение

Тестирование – исследование программы для получения информации о её качестве с точки зрения определённого круга заинтересованных лиц

Quality Assurance – процесс формирования, контроля и поддержания требуемых свойств и требуемого качества продукции

Software QA – тот же процесс, где продуктом выступает ПО

Метрики качества ПО:
- Функциональность – способность выполнять заданные функции
- Надёжность – способность ПО выполнять свои функции в заданных условиях на протяжении заданного промежутка времени и/или заданное количество раз
- Эффективность – способность ПО выполнять заданный уровень производительности в соответствии с заданными ресурсами
- Удобство использования – возможность простого и понятного использования ПО
- Удобство сопровождения – простота, с которой ПО может сопровождаться, анализироваться, контролироваться, изучаться
- Портативность – простота переноса ПО с одного окружения на другое

Виды работ в процессе тестирования:
- Test management – управление тестированием
- Test design – проектирование тестирования
- Test execution – выполнение тестирования
- Test analysis – анализ результатов тестирования

Классификация видов тестирования:
- Функциональное тестирование – проверка выполнения функциональных требований к ПО
- Нефункциональное тестирование – тестирование безопасности, производительности и других нефункциональных характеристик ПО
- Тестирование изменений – комплексное тестирование, проводимое с целью контроля качества изменений ПО

Виды функционального тестирования:
- Функциональная пригодность
- Точность
- Способность к взаимодействию
- Соответствие стандартам и правилам

Виды тестирования производительности:
- Нагрузочное тестирование – тестирование поведения ПО под нагрузкой
- Стрессовое тестирование – тестирование поведения ПО при больших нагрузках или в условиях отказа системных ресурсов
- Тестирование стабильности или надёжности – т.п. ПО при длительной работе на расчётных или чуть больших нагрузках
- Объёмное тестирование – т.п. ПО при увеличении объёма обрабатываемых или хранимых данных

Типы нефункционального тестирования:
- Тестирование установки
- Тестирование удобства пользования
- Тестирование на отказ и восстановление
- Конфигурационное тестирование
- Интернационализация и локализация
- Тестирование документации

Классификация видов тестирований с точки зрения процесса разработки:
- Unit-тестирование
- Интеграционное тестирование
- Системное тестирование
- Приёмо-сдаточные испытания

С точки зрения процесса тестирования:
- Дымовое тестирование
- Регрессионное тестирование
- Sanity-тестирование

С точки зрения способа проведения:
- Ручное тестирование – выполнение тестировщиком действий, аналогичных действиям пользователей
- Автоматизированное тестирование – тестирование при помощи тестовых скриптов, имит. действия пользователей

Плюсы автоматизированного тестирования:
- Дешевле в работе
- Надёжнее
- Быстрее
- Можно выполнять гораздо чаще ручного тестирования

Минусы:
- Требуются ресурсы на создание скриптов
- Не все виды тестирования возможно автоматизировать

Первыми кандидатами на автоматизацию обычно являются:
- Дымовое тестирование, для автоматической и оперативной проверки всех собирающихся билдов
- Регрессионное тестирование, из-за необходимости повторения одних и тех-же действий с каждой версией продукта
- Нагрузочное тестирование, так как вручную имитировать требуемую нагрузку невозможно

Артефакты тестирования:
- План тестирования – документ описывающий весь объём работ по тестированию
- Набор тест кейсов
- Дефекты и bug-репорты

План тестирования включает:
- Описание тестируемых объектов, стратегии, расписания, критериев начала и окончания тестирования
- Необходимое в процессе работы оборудование
- Требуемые специальные знания
- Оценка рисков с вариантами их разрешения

Преимущества:
- Прогнозируемая задача по тестированию
- Построение стратегии тестирования, согласованной со всей командой
- Возможность вести учёт всех требуемых ресурсов, как технических, так и человеческих
- Планирование использования ресурсов на тестирование
- Просчёт рисков, возможных при проведении тестирования

Тест-кейс должен содержать:
- Выполняемое действие
- Ожидаемый результат
- Фактический результат
- Предусловия
- Постусловия

Тест-кейсы могут быть:
- Позитивные
- Негативные

Дефект — несоответствие фактического результата выполнения программой ожидаемому результату

Баг-репорт — технический документ, содержащий в себе полное описание дефекта, включающее как информацию как в самом баге, так и о условиях его возникновения

Общая информация в баг-репорте:
- Краткое описание проблемы
- Название проекта, в котором найден дефект
- Компонент приложения, в котором возник дефект
- Версия приложения, в которой найден баг
- Серьёзность — градация степени влияния на приложение
- Приоритет — срочность исправления дефекта
- Статус дефекта в своём жизненном цикле
- Автор баг-репорта
- Назначение

Суть дефекта в баг-репорте:
- Описание:
  - Шаги воспроизведения
  - Фактический результат
  - Ожидаемый результат
- Окружение:
  - ОС
  - Разрядность
  - Версия ОС
  - Браузер и его версия
- Вложения:
  - Логи
  - Скриншоты
  - Документы, которые могут помочь воспроизвести проблему или решить её

Для быстрой оценки полноты и достаточности функциональных тестов используется покрытие тестовыми покрытия — процента всех функциональных возможностей ПО, для которых есть best cases